COMP 642 — Machine Learning Feb 1, 2022

Lecture 7: Decision Trees

Lecturer: Anshumali Shrivastava
Scribe By: Xinyang Song, Hongming Zhang, Shanli Ding, Matheus Rempel

Disclaimer: *These lecture notes are intended to develop the thought process and intuition in machine learning. The materials are not thoroughly reviewed and can contain errors.*

1 Activation Functions in Neural Networks

In perceptron, it is very useful to apply non-linear function to achieve good performance due to the following reasons:

- Linear transform is useless in multi-layper perceptron.
- The outcome of $w^T x$ is a scalar which can be any number between $[-\infty, +\infty]$. By applying non-linear function like sigmoid function, the range of outcome can be transformed to a small range to achieve better performance.

1.1 Sigmoid or Logistic Activation Function

Sigmoid function is one of the popular activation functions, which is commonly used in logistic regression.

Sigmoid Function is defined as: $\Phi(z) = \frac{1}{1+e^{-x}}$, where $z = w^T x$

The advantages of sigmoid Function is as follows:

- As we can see from Figure 1, sigmoid function is sensitive when the input is near zero, and not sensitive when the input is too far from 0. This is a benefit when people are thinking about spiking neurons in tradition.
- The output of sigmoid function is between $[0, 1]$.

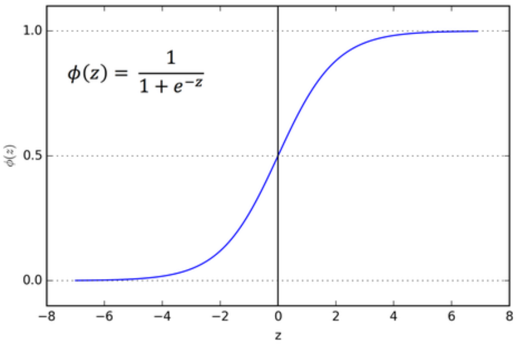


Figure 1: Sigmoid Function

1.2 Tanh Function

1.3 ReLU (Rectified Linear Unit) Activation Function

ReLU is mostly used in hidden layers in convolutional neural networks or deep learning.

2 21-40minutes

2.1 Calculation principle in neural network

The feed-forward neural network includes a full-connection feed-forward neural network and convolutional neural network. Feed-forward neural networks can be seen as a function, implement complex mapping of input spaces to output space by simple nonlinear functions.

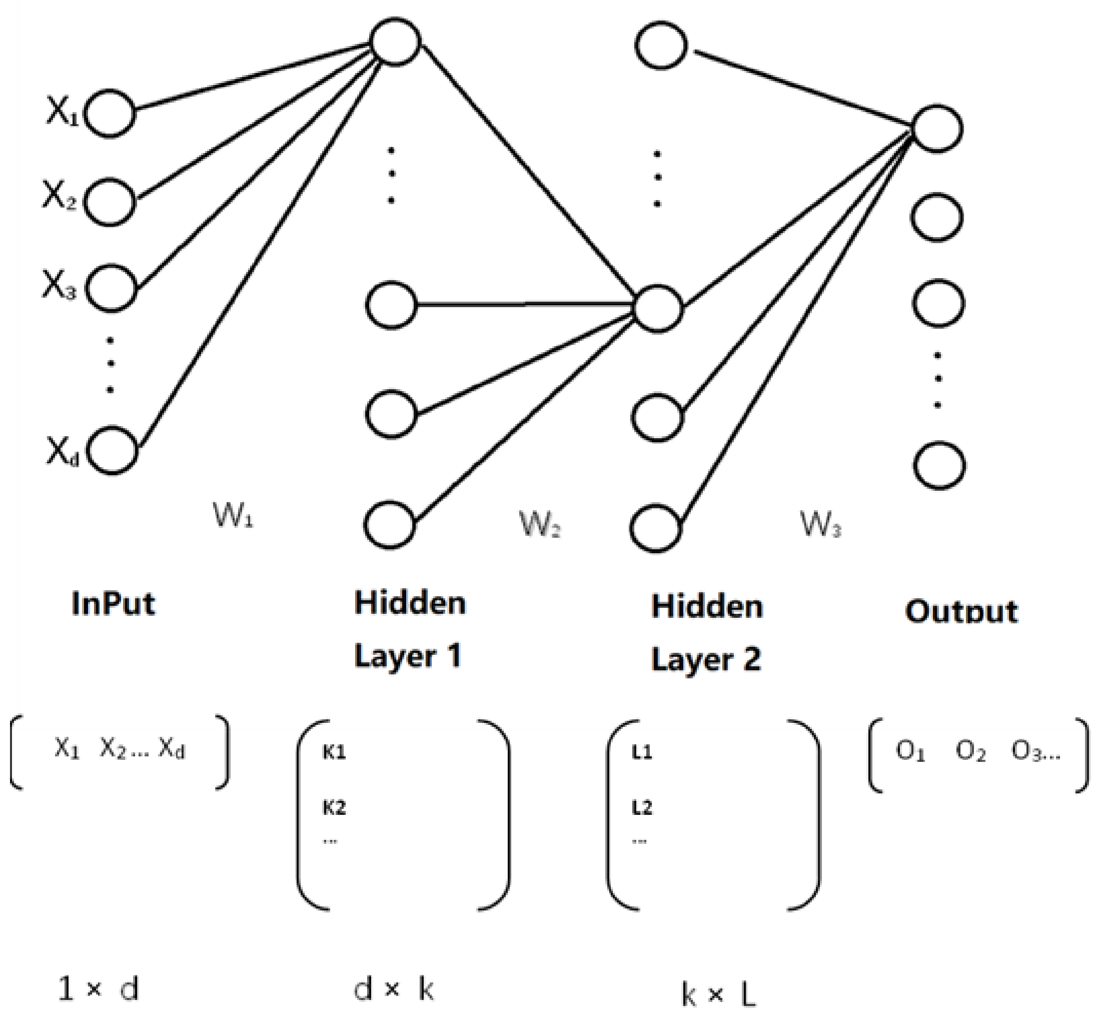


Figure 2: Simple Natural Network example

Here is an example of a neural network with 2 hidden layers, Figure 2. In the example, suppose the input data is a matrix of $1 \times D$. The first hidden layer is the matrix of $D \times K$ and the second hidden layer is the matrix of $K \times L$. the corresponding weight matrices in the three operations are W1, W2 and W3 respectively.

2.1.1 Why use activation functions

The main reason to use active function is: Increase nonlinear factors, solve the deficiencies of linear model expressive ability. When there is no neural network, the treatment of the neurons on the data is based on weight and offset.Linear transformation is simple, but limits the processing power of complex tasks.The neural network without activation functions is a linear regression model.Nonlinear transformations made by activation functions allow neural networks to handle very complex tasks.

For example, in Figure 2, assuming that the input feature x is only one feature, then the two hidden layer neural network (each hidden layer is single node) output $O = w_3 w_2 w_1 x$ is a linear function. Therefore, if the activation function is linear, how many layers the neural network has does not make sense, and the most basic linear regression is exactly the same, the representation of the results of the above figure and $O = wx$ are the same.

2.1.2 Calculation principle

The calculation process of the neural network is the overall output of the input data to obtain the final output through the matrix operation.Taking the neural network in Example as an example, the calculation process of the neural network is as follows:

* Step1: Enter the input matrix of $1 \times D$.

* Step2: Select the appropriate nonlinear activation function, apply on every neuron.

* Step3: After the input matrix is activated and did Matrix Operations the hidden layer 1, get the output matrix of $1 \times K$.

* Step4: STEP3 gets the matrix of $1 \times K$ is activated and with the hidden layer 2 for matrix operation, get the output matrix of $1 \times L$.

* Step5: STEP4 Get a matrix operation after the $1 \times L$ matrix is activated, resulting in the output matrix O.

* STEP6: The loss function f = (R, O) for the results (R, O) is calculated (R is the correct result, O is a prediction result), optimizing the neural network.

2.2 Back Propagation Concept

After the forward propagation, because there is an error between the output result and the actual result, the error between the estimated value and the actual value is calculated by using the loss function, and the error is back propagated from the output layer to the hidden layer to the input layer. This process is called back propagation.

The steps of back propagation can be simply understood as the following steps:

* Step 1: forward propagation

* Step 2: back propagation In the process of back-propagation, the values of parameters such as weight are adjusted according to the error to continuously make the final output better

* Step 3: repeat step 1 and step 2 until you get the best results.

3 Feature Learning

Feature learning is a process that can help our model understand important and core features for either feature recognition in object detection/image classification challenges or clustering groups from unlabeled datasets. In other words, machine learning engineers prefer to use inputs that are suitable with such operations or tasks, however, eliminating the irrelevant noise inputs from the modeling process. There are two types of learning processes, supervised and unsupervised that are, simply speaking, labeled configured data inputs and unlabeled unknown data inputs. In the real world, there are lots of different ways to do a feature learning/engineering. For example, the most basic approach to understand the numeric data inputs' status is to inspect statistical conditions, such that mean, standard deviation, variance, minimum, and maximum, etc. These basic information can help develop an intuition understanding of the data inputs. Furthermore, there are more advanced approaches, for example, *Principle Component Analysis(PCA)*[2], *Multidimensional Scaling(MDS)*[3], *Sammon Mapping Algorithm*[4], etc.

3.1 Feature Engineering Example

To be more specifically, we can take the Boston Housing Price challenge[5] as a great example for applying feature learning before building the prediction model. There are up to 14 tags/features for 1 house environment. It is a burdensome work to build up a model that uses all 14 features and learns a trend , because there might be some of them that are irrelevant to model's goal. To get a sense of what our dataset looks like, we can perform a basic statistical operations and, also, plot a $n \times m$ correlation matrix for an explicit view of data inputs which n and m are for number of features in the data set respectively. Among all 14 features in data inputs, we can make a conclusion that there are 4 specific features are strongly related to the learning task by reference the feature inspection outputs from Figure 3. There are more to try out with this data set to gain a understanding of the importance of feature engineering, and there is an easy access for sklearn library.

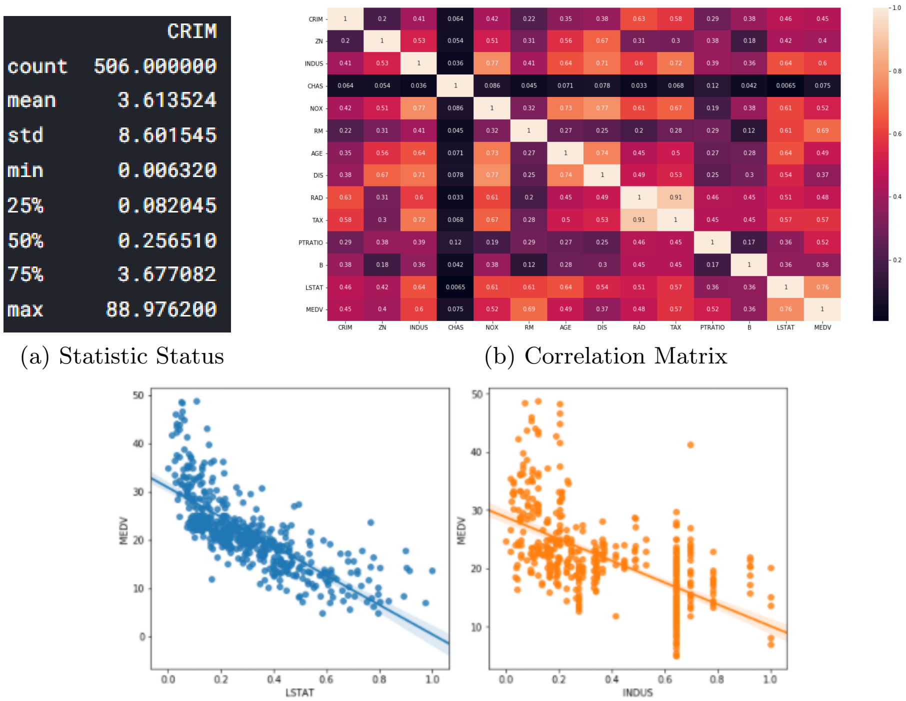
3.2 Principle Component Analysis

Principal component analysis, also known as PCA, tries to reduce datasets' dimensions, and at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance.[2] In the equation(1) shows how to calculate the quality of *k-dimension* data sets. π stands for the quality indicator, λ is a Lagrange multiplier, and $tr(S)$ is the trace of matrix S.

$$\pi_i = \frac{\lambda_i}{\sum_{i=1}^{k} \lambda_i} = \frac{\lambda_i}{tr(S)}\text{‘} \tag{1}$$

3.3 Multidimensional Scaling

Multidimensional scaling algorithm(MDS) is intended to signify high dimensional data set in a low dimensional space with not distortion each data positions much. Such dimensional reduction also can help engineers examine hidden data relationships and importance to the goal. There are different types of MDS nowaday, such as *Classical multidimensional scaling* or *Principal Coordinates Analysis (PCoA)*[1], *Metric multidimensional scaling (mMDS)*[3], *Non-metric multidimensional scaling (nMDS)*[3].



```
              CRIM
count   506.000000
mean      3.613524
std       8.601545
min       0.006320
25%       0.082045
50%       0.256510
75%       3.677082
max      88.976200
```

(a) Statistic Status

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	0.2	0.41	0.064	0.42	0.22	0.35	0.38	0.63	0.58	0.29	0.38	0.46	0.45
ZN	0.2	1	0.53	0.054	0.51	0.31	0.56	0.67	0.31	0.3	0.38	0.18	0.42	0.4
INDUS	0.41	0.53	1	0.036	0.77	0.41	0.64	0.71	0.6	0.72	0.39	0.36	0.64	0.6
CHAS	0.064	0.054	0.036	1	0.086	0.045	0.071	0.078	0.033	0.068	0.12	0.042	0.0065	0.075
NOX	0.42	0.51	0.77	0.086	1	0.32	0.73	0.77	0.61	0.67	0.19	0.38	0.61	0.52
RM	0.22	0.31	0.41	0.045	0.32	1	0.27	0.25	0.2	0.28	0.29	0.12	0.61	0.69
AGE	0.35	0.56	0.64	0.071	0.73	0.27	1	0.74	0.45	0.5	0.27	0.28	0.64	0.49
DIS	0.38	0.67	0.71	0.078	0.77	0.25	0.74	1	0.49	0.53	0.25	0.3	0.54	0.37
RAD	0.63	0.31	0.6	0.033	0.61	0.2	0.45	0.49	1	0.91	0.46	0.45	0.51	0.48
TAX	0.58	0.3	0.72	0.068	0.67	0.28	0.5	0.53	0.91	1	0.45	0.45	0.57	0.57
PTRATIO	0.29	0.38	0.39	0.12	0.19	0.29	0.27	0.25	0.46	0.45	1	0.17	0.36	0.52
B	0.38	0.18	0.36	0.042	0.38	0.12	0.28	0.3	0.45	0.45	0.17	1	0.36	0.36
LSTAT	0.46	0.42	0.64	0.0065	0.61	0.61	0.64	0.54	0.51	0.57	0.36	0.36	1	0.76
MEDV	0.45	0.4	0.6	0.075	0.52	0.69	0.49	0.37	0.48	0.57	0.52	0.36	0.76	1

(b) Correlation Matrix

(c) Linear Relationship

Figure 3: Feature Inspection

4 Decision Trees

4.1 Decision Tree Background

Data Trees are effectively a program which takes data as an input. Ultimately speaking, every time a decision is made, an if then else statement is taken in order to predict the output of an input various features. Whilst at times it is thought that decision trees are elementary and hence not useful for use. However, some of the largest companies in the world (ie Google and Facebook) often use decision trees due to their extreme accuracy. The biggest issue with decision trees is that there is the risk that can effectively lead to a memorization model (zero error).

4.2 Decision Tree Setup

In order to set up a tree, features are usually broken into binary decisions (whilst they can be broken into more than two options, for this specific class, the case of a binary feature option was proposed). Once established, these steps are repeated until a leaf node is reached (a node which has no more children). At this point, the data is considered, and the more likely outcome is deemed to be the output as seen below in Figure 4.

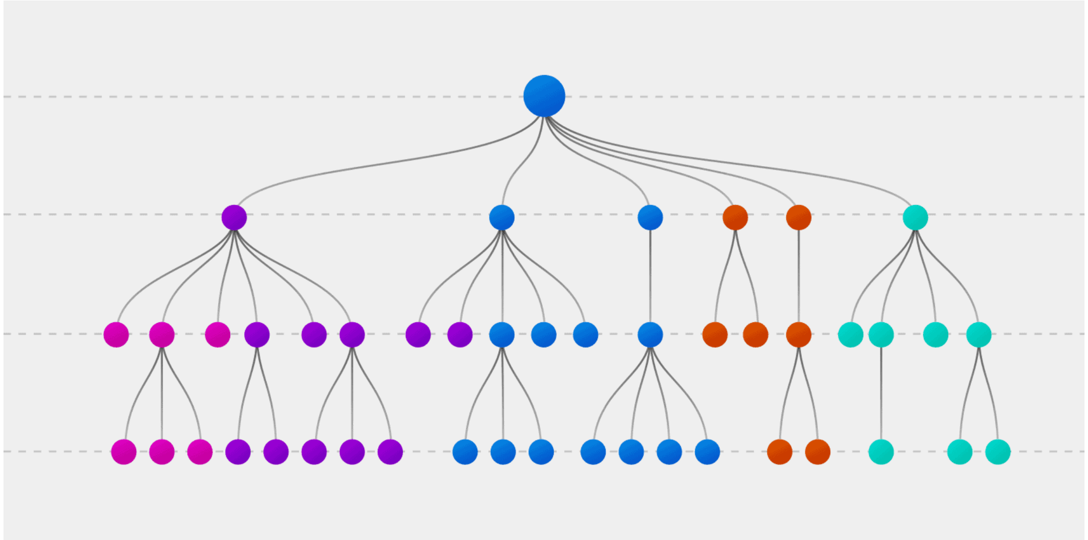


Figure 4: Example decision tree with 20 leaf nodes

4.3 Industry Use

In industry, there are two primary uses decisions trees, mass indexing and allowing for explanations off algorithmic decisions. The former was not discussed by end of the class (refer to lecture 8 for this), but the latter was briefly mentioned. In the modern day, often times, businesses cannot get away with simply providing the decision of the algorithm without an adequate rationale. The example given was; image someone applies for a loan in the bank, and the decision was to reject their application. Using decision trees, one can intuitively go back to the structure and give both a rationale and future recommendations. For example, if your income was 5000 dollars more per year, or if you had 3000 dollars less debt, you would have been approved for this loan. Ultimately, a much more valid answer than: ”Sorry, you have been rejected for the loan.”

References

[1] John C. Gower. *Principal Coordinates Analysis*, pages 1–7. John Wiley Sons, Ltd, 2015.

[2] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[3] A. Mead. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 41(1):27–39, 1992.

[4] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[5] U.S. Census Service. Boston housing price. http://lib.stat.cmu.edu/datasets/boston. Accessed: 2022-02-06.